

PAPER • OPEN ACCESS

Comparisons of performance between quantum-enhanced and classical machine learning algorithms on the IBM Quantum Experience

To cite this article: P V Zahorodko *et al* 2021 *J. Phys.: Conf. Ser.* **1840** 012021

View the [article online](#) for updates and enhancements.

You may also like

- [Quantum information processing with superconducting circuits: a review](#)
G Wendin
- [Verifying commuting quantum computations via fidelity estimation of weighted graph states](#)
Masahito Hayashi and Yuki Takeuchi
- [Quantum supremacy of many-particle thermal machines](#)
J Jaramillo, M Beau and A del Campo



245th ECS Meeting • May 26-30, 2024 • San Francisco, CA

Don't miss your chance to present!

Connect with the leading electrochemical and solid-state science network!

Deadline Extended: December 15, 2023

Submit now!



Comparisons of performance between quantum-enhanced and classical machine learning algorithms on the IBM Quantum Experience

P V Zahorodko¹, S O Semerikov^{1,2,3}, V N Soloviev¹, A M Striuk², M I Striuk² and H M Shalatska²

¹ Kryvyi Rih State Pedagogical University, 54 Gagarin Ave., Kryvyi Rih, 50086, Ukraine

² Kryvyi Rih National University, 11 Vitalii Matusevych Str., Kryvyi Rih, 50027, Ukraine

³ Institute of Information Technologies and Learning Tools of NAES of Ukraine, 9 M. Berlynskoho Str., Kyiv, 04060, Ukraine

E-mail: semerikov@gmail.com

Abstract. Machine learning is now widely used almost everywhere, primarily for forecasting. In the broadest sense, the machine learning objective may be summarized as an approximation problem, and the issues solved by various training methods can be reduced to finding the optimal value of an unknown function or restoring a function. At the moment, we have only experimental samples of quantum computers based on classical-quantum logic, when quantum gates are used instead of ordinary logic gates, and probabilistic quantum bits are used instead of deterministic bits. Namely, the probabilistic nature problems that provide for the determination of a certain optimal state from a large set of possible ones on which quantum computers can achieve “quantum supremacy” – an extraordinary (by many orders of magnitude) reduction in the time required to solve the task. The main idea of the work is to identify the possibility of achieving, if not quantum supremacy, then at least a quantum advantage when solving machine learning problems on a quantum computer.

1. Introduction

Traditionally, quantum computing is defined as a type of nonclassical computing that operates on the quantum state of subatomic particles, which represent information as elements denoted as quantum bits (qubits). A qubit can represent all possible values simultaneously (superposition) until read. Qubits can be linked with other qubits, a property known as entanglement. Quantum algorithms manipulate linked qubits in their undetermined (entangled) state, a process that can address problems with vast combinatorial complexity [7], reaching “quantum supremacy”.

Identifying potential applications for quantum computing, Kasey Panetta points out that they “will be narrow and focused, as general-purpose quantum computing will most likely never be economical” [13]. In his opinion, quantum computing could enable breakthroughs by machine learning, finance, healthcare, creation of new materials, artificial intelligence (which requires 100s – 1000s qubits), chemistry and biochemistry (100-200 qubits). In particular, for finance, quantum computing could enable faster, more complex Monte Carlo simulations (for example, trading, trajectory optimization, market instability, price optimization and hedging strategies) and machine learning methods, which in



the general case are reduced to problems of finding the extremum of a multidimensional function along the nonlinear response surface.

Currently, computing devices capable of performing quantum computing (quantum computers) are available for consumers of computing services using the QCaaS (quantum computing as a service) model. As of June 2020, the maximum number of qubits available for simultaneous use does not exceed 60, which is significantly less than the number required to achieve “quantum supremacy”. This raises the problem of investigating the possibilities of quantum programming for machine learning tasks implementation, namely, the use of machine learning algorithms, implemented by the quantum programming language, to analyze traditional data and compare the performance of quantum and von-neumanns implementations at the present stage of their development.

The aim of the research is to carry out a comparative analysis of the implementation of quantum-enhanced machine learning algorithms in the quantum programming language.

To achieve the goal of the study, the following tasks were set:

1. Analyze Quantum Software Engineering tools in order to select a tool that is appropriate for the implementation of machine learning tasks.
2. Consider quantum-enhanced machine learning algorithms.
3. Conduct a comparative analysis of the efficiency of quantum-enhanced and classical machine learning algorithms.

2. Fundamentals of Quantum Software Engineering

2.1. Basic research concepts

Quantum computer is a computing device using quantum-mechanical phenomena (superposition, entanglement, etc.) for data transmission and processing.

Quantum programming is a software development process for quantum computer.

«Classical» applications of quantum computers (by Richard Feynman) – modeling complex [many-particle physical] systems: Zalka and Wiesner’s algorithm.

«New» applications of quantum computers are tasks that require enumerating a large number of options: Grover’s algorithm (general task), Shor’s algorithm (factorization), Abrams and Lloyd’s algorithm (identification of periodic properties), etc.

Quantum machine learning is an application of machine learning algorithms for quantum data analysis.

Quantum-enhanced machine learning is the use of machine learning algorithms implemented in the quantum programming language for the analysis of traditional data.

Software Engineering is a systematic application of engineering approaches to the design, implementation, testing and documenting of software.

2.2. Concept of Quantum Software Engineering

The first systems presentation of the Quantum Software Engineering concept was made by John Clark and Susan Stepney in 2002 [4]. Researchers believe that quantum computing cannot be effectively implemented in the traditional computer Von Neumann architecture, the mathematical model of which is the Turing machine. The authors [3] refer to the main challenges that Quantum Software Engineering will face in 2020:

- the question of what a quantum programming language should be – an extension of traditional languages, a logical programming language in a low-level programming language or a language that implements a new paradigm;
- the need to develop compilers for quantum programming languages;
- the need to develop new quantum algorithms and define the classes of traditional algorithms that can be quantised;
- feasibility of developing quantum computer simulators for use on traditional computer systems;

– despite the fact that quantum execution is in principle unobservable, debugging and testing techniques are necessary for quantum programming languages;

– quantum algorithms require visualization for their understanding, design, and implementation.

The criteria and success indicators of Quantum Software Engineering proposed by John Clark and Susan Stepney are summarized in table 1.

Table 1. The criteria and success indicators of Quantum Software Engineering (according to [4]).

Criteria	Indicators
It arises from scientific curiosity about the foundation, the nature or the limits of a scientific discipline	Quantum computation has broadened the fundamental limits of computer science and software engineering
The ability to create new engineering solutions	The physical infrastructure is constantly evolving, each solution is new
Technological continuity	The existence of high level languages and development techniques that can be used by computer scientists and software engineers with only the same style of training they receive today (so, no need to teach the fundamentals of quantum mechanics to all)
Research community support	Support for all interested in new computing paradigms and new levels of computing power
International character of research	This is a new fundamental area of software engineering
It is generally comprehensible, and captures the imagination of the general public, as well as the esteem of scientists in other disciplines	It is not generally understood, but is known for its worldwide interpretation
The problem has a long-standing statement, but has not yet been resolved	Formulated by Richard Feynman in the late 1970s.
It promises to go beyond what is initially possible, and requires development of understanding, techniques and tools unknown at the start of the project	Problems exist on every level, from developing a whole new conceptual paradigm, to building intellectual and simulation tools
It calls for planned co-operation among identified research teams and communities	Research is needed in a number of areas (languages, algorithms, tools, simulation, visualisation, etc.)
It encourages and benefits from competition among individuals and teams, with clear criteria on who is winning, or who has won	There need not be a single “winner”, diversity of solutions should be encouraged, as in classical software engineering, to be applicable to a range of application domains
It decomposes into identified intermediate research goals, whose achievement brings scientific or economic benefit, even if the project as a whole fails	There are several components of the problem that can be explored in parallel
It will lead to radical paradigm shift	Quantum computing is a radical paradigm shift

In 2020, Quantum Software Engineering includes such components [19]:

- Paradigms for developing quantum software
- Quantum software design
- Quantum software testing
- Quantum software verification
- Quantum software coding practices
- Quantum software reuse
- Quantum software experimentations
- Quantum software execution
- Industrial applications
- Empirical evaluations

In February 2020, at QANSWER 2020: 1st International Workshop on the QuANtum SoftWare Engineering & pROgramming, the Talavera Manifesto for Quantum Software Engineering and Programming [12] was adopted, containing a set of principles and commitments:

Quantum Software Engineering

- *is agnostic regarding quantum programming languages and technologies;*
- *embraces the coexistence of classical and quantum computing,* and advocates the use of reengineering techniques to integrate new quantum algorithms with the existing classical information systems. Reverse engineering techniques are also needed to parse and abstract quantum program information that is to be integrated into classical programs;
- *supports the management of quantum software development projects,* delivering quantum software that fulfils the initial business goal and requirements, while at the same time ensuring that quality, time, and cost constraints are being properly observed; methodologies for developing quantum programs must be created or adapted from the existing ones; effort estimation methods for quantum software development need to be provided as well;
- *considers the evolution of quantum software:* quantum software should be maintained and evolved from inception to removal, and quantum software evolution must be handled throughout the whole quantum software lifecycle;
- *aims at delivering quantum programs with desirable zero defects:* It is in charge of defining and applying testing and debugging techniques to quantum programs in such a way that most defects can be detected and solved before the program is released;
- *assures the quality of quantum software:* quality management for both process and product are essential if quantum software with expected quality levels is to be produced; since we cannot improve what we cannot measure, new metrics for quantum programs and quantum processes have to be developed;
- *promotes quantum software reuse,* helping development teams to share, index, and find quantum software that can be reused: this requires study of design and architectural patterns for quantum programs, facilitate technical communication, and work on creating libraries of reference examples and application demonstrations;
- *addresses security and privacy by design:* quantum information systems must be secure and guarantee the privacy of data and of users from the initial phases of quantum software development, i.e., by design;
- *covers the governance and management of software:* managers should be aware of the particular processes, organizational structures, principles, policies and frameworks, information, culture, ethics and behaviour, people, skills and competences, as well as the services, infrastructure and applications that are associated with quantum software and that are (or should be) provided by organizations.

The authors of the manifesto separately appeal to educators with a request to integrate quantum software engineering in curricula within the existing software engineering degrees and/or courses in this or other disciplines, and clearly specify which competences and skills are required for future quantum software engineers [15].

2.3. *Quantum Software Engineering tools*

The execution of quantum programs on personal computer equipment is difficult to access due to its lack of prevalence, so for more than a quarter-century, quantum simulators – software tools that simulate quantum circuits – have been the main means of their execution. The first mention of QCaaS (Quantum Computing as a Service) occurs only in 2015 in the article [20] by Mijanur Rahaman and Md. Masudul Islam.

The world's largest QCaaS providers:

- D-Wave Systems Inc. (Canada) – SDK Ocean [6] (Python, C++);
- International Business Machines Corporation (USA) – SDK ProjectQ [22] (Python), Qiskit [17] (Python);
- Cambridge Quantum Computing Limited (Great Britain) – SDK tket [2] (Python);
- QC Ware, Corp. (USA) – SDK Forge (Python);
- StationQ - Microsoft (USA) – SDK LIQI|> [11] (F#), Microsoft Quantum Development Kit [10] (F#);
- Rigetti Computing (USA) – SDK Forest [21] (Python).

Thus, the main programming language for cloud access to quantum computing is Python. Another criterion for choosing a QCaaS vendor is computing power, measured in qubits. This indicator is the largest in D-Wave Advantage – 5000 (in clusters of 8) qubits based on quantum annealing, which narrows the scope of its application to solving optimization problems, which boil down to finding the ground state for a set of spins. For universal quantum computers on quantum circuits, the number of qubits is significantly lower and today (June 2020) is the highest in IBM Q 53 (53 qubits) and Google Bristlecone (72 qubits). Unfortunately, Google's Quantum Computing Playground [19] is a browser-based quantum simulator, and there is no open cloud access to Google's Bristlecone. For cloud access to IBM Q, you can use both their library – Qiskit, and a third-party – ProjectQ. Considering that the highest level of specialization is provided by its own SDK, Qiskit was chosen for further work.

3. Quantum-enhanced machine learning

3.1. *Quantum models of machine learning*

Srinivasan Arunachalam and Ronald de Wolf [1] offer three main quantum learning models:

1. *Quantum exact learning* based on membership queries to find the most accurate unknown function (quantum approximation problem). The efficiency of quantum algorithms in relation to classical ones in this case depends on how the learning efficiency is measured. If the measure of efficiency is the training time, then there are such classes of functions for which quantum algorithms are much faster than classical ones, assuming that the queries implementation in a quantum superposition is possible.

2. *Quantum Probably Approximately Correct (PAC) learning* to find an unknown function over a set of samples (quantum supervised learning). The difference between quantum PAC learning and classical learning is that the dataset can be in a state of quantum superposition.

3. *Quantum agnostic learning* to search for the $(n + 1)$ -th bit, which is a continuation of a sequence with n bits (quantum prediction task).

The authors point to three types of complexity that arise when applying quantum learning models [1]:

a) query complexity of quantum exact learning: the number of quantum membership queries needed to exactly learn a target concept can be polynomially smaller than the number of classical membership queries, but not much smaller than that;

b) sample complexity: for the distribution-independent models of PAC and agnostic learning, quantum examples give no significant advantage over classical random examples: for every concept class, the classical and quantum sample complexities are the same up to constant factors. In contrast, for some fixed distributions (e.g., uniform) quantum examples can be much better than classical examples;

c) time complexity: there exist concept classes that can be learned superpolynomially faster by quantum computers than by classical computers, for instance based on Shor's or Simon's algorithm.

In the case of applying quantum machine learning models to the analysis of traditional data, we are talking about quantum-enhanced machine learning. Frank Phillipson [11] defines three main benefits of quantum machine learning:

- improving runtime (for example with a quantum hybrid Helmholtz machine);
- learning capacity improvements (for example with a quantum Hopfield neural network);
- learning efficiency improvements: less training information or simpler models needed to produce the same results or more complex relations can be learned from the same data.

Various methods can be applied to increase the efficiency of training, one of which is variational quantum circuits – VQC [14].

Evidence of the intensity of quantum-enhanced machine learning development is the fact that the systematic review of the problem in 2016, carried out by Peter Wittek in [23], today (June 2020) is already considered as a classic, and that is indicated by the co-author in a new review [5].

Vedran Dunjko and Peter Wittek also highlight such perspective directions in the development of quantum machine learning in general:

a) supervised and unsupervised learning: continuous-variable quantum neural networks, quantum convolutional neural networks, quantum algorithms for feedforward neural networks, Bayesian deep learning, sublinear quantum algorithms for training linear and kernel-based classifiers;

b) reinforcement learning: quantum algorithms for solving dynamic programming problems (including hidden quantum Markov models), quantum gradient estimation.

The authors conclude that “the entire field of “genuinely quantum” machine learning (where the data itself is quantum) is still finding its right place and full recognition. Perhaps as quantum technologies mature, and problems of quantum learning become genuinely practical, the field will crystallize and grow. ... In summary, QML [quantum machine learning] is diverse, growing, inclusive, and it is rich in open questions. ... Capturing all the QML trends, which will in the end be central is, for the time being, an impossible task – and, in a way, this is the key message of this note” [5].

3.2. *An overview of quantum-enhanced machine learning tools in Qiskit*

Qiskit provides the ability to develop quantum software both at the quantum circuits level using OpenQASM [16] and at a high level of abstraction using Python in a Jupyter notebook. The main components of the library are:

- quantum circuits modeling tools (Terra);
- implementation of standard quantum algorithms (Aqua – Algorithms for QUantum Applications), in particular, for solving optimization tasks;
- cloud quantum computing tools (Aer);
- tools for simulating quantum noise (Ignis).

Aqua includes modules for research in finance (qiskit.finance), machine learning (qiskit.ml), optimization (qiskit.optimization) and chemistry (qiskit.chemistry) [16].

The machine learning module contains standard datasets and ways to access custom. Various optimization algorithms can be used to process them:

ADMMOptimizer – an implementation of the ADMM-based heuristic (ADMM – alternating direction method of multipliers);

CobylaOptimizer – the SciPy COBYLA optimizer (COBYLA – Constrained Optimization BY Linear Approximation);

CplexOptimizer – the CPLEX optimizer for linear, integer and quadratic programming tasks;

GroverOptimizer – uses Grover Adaptive Search (GAS) to find the minimum of a QUBO function (QUBO – quadratic unconstrained binary optimization);

MinimumEigenOptimizer – minimum eigen solvers;

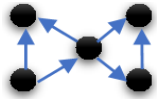
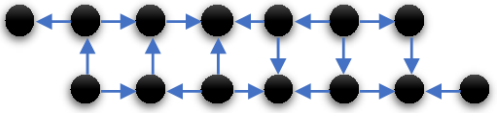
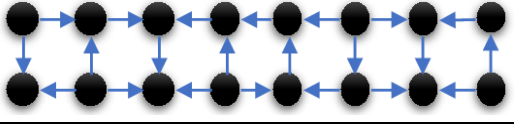

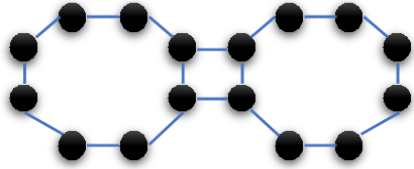

RecursiveMinimumEigenOptimizer – a meta-algorithm that applies a recursive optimization.

The `qiskit.aqua.components.optimizers` module offers a set of algorithms for local (Analytic quantum gradient descent optimizer, constrained optimization by linear approximation optimizer, Nelder-Mead optimizer, Nakanishi-Fujii-Todo algorithm, Powell optimizer, truncated Newton optimizer, etc.) and global optimizations (controlled random search with local mutation optimizer, evolutionary optimizer, etc.). It is advisable to use quantum support vector machine (QSVM) and variational quantum classifier (VQC) algorithms to solve classification tasks.

4. Experimental verification of the efficiency of quantum-enhanced machine learning algorithms

When implementing quantum computing on real architectures, in addition to the classical decoherence problem, which significantly limits the running time of quantum algorithms (70-150 μ s for a 20-qubit IBM Q System One with a maximum number of simultaneously entangled qubits equal to 6), it is also necessary to take into account error reading the results. The authors [9] provide the following values for quantum computers accessible via QCaaS (table 2).

Table 2. Characteristics of common quantum computers (according to [12]).

Machine	Qubits	2Q gates	Coherence time (μ s)	1Q error, %	2Q error, %	Read output, %	Qubit topology
IBM Q5 Tenerife	5	6	40	0,2	4,76	6,21	
IBM Q14 Melbourne	14	18	30	1,19	7,95	9,09	
IBM Q16 R�schlikon	16	22	40	0,22	7,14	4,15	
Rigetti Agave	4	3	15	3,68	10,8	16,37	
Rigetti Aspen1	16	18	20	3,43	8,92	5,56	
Rigetti Aspen3	16	18	20	3,79	5,37	6,65	
UMD Trapped Ion	5	10	1,5 \cdot 10 ⁶	0,2	1,00	0,6	

Thus, at the current state of quantum technologies development, it is necessary to use such data sets, the processing steps of which by quantum-enhanced machine learning algorithms correspond to the requirements of the chosen platform – IBM Q.

Machine learning was performed on the wine and breast_cancer datasets using the Qiskit (quantum-enhanced machine learning) and sklearn (classical machine learning) libraries. Testing was performed on a local two-qubit quantum simulator with 8 GB of RAM and servers ibmqx2 (2 qubits),

ibmq_16_melbourne (16 qubits), ibmq_vigo, ibmq_london and ibmq_burlington (all 5 qubits). The machine learning procedure was repeated 6 times for each dataset. The results are presented in table 3.

Table 3. Experimental verification of the efficiency of machine learning on quantum and traditional architectures.

Server	Time of quantum-enhanced machine learning (s)		Time of classical machine learning (s)	
	wine	breast_cancer	wine	breast_cancer
ibmqx2	24.8	18		
	24.7	18.1		
	24.5	17.9		
	24.6	18.1		
	24.8	18.1		
	24.7	17.8		
ibmq_16_melbourne	25	18.9		
	25.2	18.7		
	25.2	10.2		
	25.1	10.2		
	25.2	18		
	24.8	18.9		
ibmq_vigo	28.3	20.4	0.021	0.027
	28.3	20.4	0.021	0.027
	28.1	20.4	0.026	0.026
	28	20.7	0.026	0.027
	26.9	20.4	0.026	0.027
	28.3	20.7	0.026	0.026
ibmq_london	26.1	19.4		
	26.2	19.3		
	26.5	19.6		
	26.5	19.5		
	26.3	19.4		
	26.2	19.7		
ibmq_burlington	27.3	20.4		
	27.1	20.4		
	26.7	20.3		
	27.1	20.5		
	27	20.3		
	27.1	20.1		
local quantum simulator	111	22.2	0.190	0.021
	105.1	23.3	0.020	0.023
	102.8	32.7	0.028	0.021
	103.5	31.7	0.021	0.021
	111.2	22.4	0.020	0.026
	99.9	22.1	0.022	0.021

When using sklearn on IBM Q Experience, there was no way to determine on which server the execution was occurring – multiple repetitions of tests at different times of the day did not lead to a significant change in the results.

The code for all types of tests is presented in appendices A, B, C.

Analysis of Table 3 allows us to conclude that at the current stage of quantum technologies development traditionally machine learning provides greater performance than quantum-enhanced. At the same time, quantum-enhanced machine learning algorithms turned out to be inversely sensitive to the complexity of the dataset: training on a more complex dataset breast_cancer (30 inputs, 2 outputs, 569 elements) was performed at a higher speed than training on a less complex data set wine

(13 inputs parameters, 3 outputs, 178 elements), while a direct relationship was observed for classical machine learning, confirmed by [2] and other sources. The results of the analysis give an opportunity to make the assumption that it is advisable to apply quantum-enhanced machine learning to datasets with a large input dimension, the assumed value for which is the probability of choosing one of two sets of classes – such classes are effectively worked out by single-qubit systems.

5. Conclusions

1. The core of Quantum Software Engineering is quantum programming – the process of developing programs for a quantum computer: a computing device that uses the phenomena of quantum mechanics to process data. Due to the low level of availability of such devices, it is advisable to access them under QCaaS model (quantum computing as a service). The conducted review of Quantum Software Engineering tools provided an opportunity to single out their main classes (quantum simulators, libraries, visualizers and cloud quantum services) and recommend using IBM Q as a hardware platform for quantum computing, Qiskit as a library of quantum algorithms, Python as a programming language and IBM Quantum Experience as QCaaS Provider.

2. The use of machine learning algorithms for the analysis of quantum data can be described by three quantum machine learning models (quantum exact learning, quantum Probably Approximately Correct learning and quantum agnostic learning), in the application of which there are three types of difficulties associated with the query complexity of quantum exact learning, quantum the intricacy of datasets and the sensitivity of quantum algorithms to them. A prospective direction in the machine learning development is the use of quantum learning models for analyzing traditional data, the implementation of which in Qiskit Aqua 0.7.3 is still a limited solution to classification tasks.

3. The results of an experiment using a variational quantum classifier on two datasets showed that at the current stage of quantum technologies development, classical machine learning provides greater performance than quantum-enhanced machine learning. At the same time, the use of quantum-enhanced machine learning algorithms for the binary classification tasks, even with a high dimension of the input data, gives a significant (several times) acceleration compared to the ternary classification tasks, while when using classical machine learning, the execution time increased depending on the volume dataset and their dimensions. The analysis of the experimental results provides an opportunity to make an assumption that quantum-enhanced machine learning is advisable to apply to datasets with a large input dimension, the assumed value for which is the probability of choosing one of two sets of classes – such classes are efficiently processed by one-qubit systems.

Prospects for further research are in a systematic study of the capabilities of Quantum Software Engineering and its applications to solving forecasting problems.

Appendix A. Code for quantum-enhanced machine learning on the wine dataset (ibmq_burlington server, 5 qubits)

```
# Importing standard Qiskit libraries and configuring account
from qiskit import QuantumCircuit, execute, Aer, IBMQ
from qiskit.compiler import transpile, assemble
from qiskit.tools.jupyter import *
from qiskit.visualization import *
from qiskit import BasicAer
from qiskit.aqua import QuantumInstance, aqua_globals
from qiskit.aqua.algorithms import VQC
from qiskit.aqua.components.optimizers import COBYLA
from qiskit.aqua.components.feature_maps import RawFeatureVector
from qiskit.ml.datasets import wine
from qiskit.circuit.library import TwoLocal
import time

# Loading your IBM Q account(s)
provider = IBMQ.load_account()
```

```

from qiskit import BasicAer
from qiskit.aqua import QuantumInstance, aqua_globals
from qiskit.aqua.algorithms import VQC
from qiskit.aqua.components.optimizers import COBYLA
from qiskit.aqua.components.feature_maps import RawFeatureVector
from qiskit.ml.datasets import wine
from qiskit.circuit.library import TwoLocal
import time

seed = 1376
aqua_globals.random_seed = seed

# Use Wine data set for training and test data
feature_dim = 4 # dimension of each data point
_, training_input, test_input, _ = wine(training_size=12,
                                         test_size=4, n=feature_dim)

instance = QuantumInstance(provider.get_backend('ibmq_burlington'),
                           shots=1024, seed_simulator=seed, seed_transpiler=seed,
                           skip_qobj_validation=True)
feature_map = RawFeatureVector(feature_dimension=feature_dim)
start_time = time.time()
vqc = VQC(COBYLA(maxiter=100),
          feature_map,
          TwoLocal(feature_map.num_qubits, ['ry', 'rz'], 'cz', reps=3),
          training_input, test_input)
result = vqc.run(instance)

print('Testing accuracy: {:.2f}'.format(result['testing_accuracy']))
print(result)

print("--- %s seconds ---" % (time.time() - start_time))

```

Appendix B. Code for quantum-enhanced machine learning on breast_cancer dataset (local quantum emulator, 5 qubits)

```

from qiskit import BasicAer
from qiskit.aqua import QuantumInstance, aqua_globals
from qiskit.aqua.algorithms import VQC
from qiskit.aqua.components.optimizers import COBYLA
from qiskit.aqua.components.feature_maps import RawFeatureVector
from qiskit.ml.datasets import breast_cancer
from qiskit.circuit.library import TwoLocal
import time
import random

seed = 1376
aqua_globals.random_seed = seed

# Use Wine data set for training and test data
feature_dim = 2 # dimension of each data point
X_train = []
Y_train = []
_, training_input, test_input, _ = breast_cancer(training_size=12,
                                                  test_size=4, n=feature_dim)
instance = QuantumInstance(BasicAer.get_backend('statevector_simulator'),
                           shots=1024, seed_simulator=seed, seed_transpiler=seed)

```

```

feature_map = RawFeatureVector(feature_dimension=feature_dim)
start_time = time.time()
vqc = VQC(COBYLA(maxiter=100),
          feature_map,
          TwoLocal(feature_map.num_qubits, ['ry', 'rz'], 'cz', reps=3),
          training_input, test_input)
result = vqc.run(instance)

print('Testing accuracy: {:.2f}'.format(result['testing_accuracy']))
print(result)

print("--- %s seconds ---" % (time.time() - start_time))

```

Appendix C. Classical machine learning code on breast_cancer dataset

```

import numpy as np
from sklearn.datasets import load_wine
from sklearn.svm import SVC
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris, load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.svm import LinearSVC, SVC
from sklearn.metrics import classification_report
from sklearn.model_selection import GridSearchCV
import time
import pandas as pd

# Load data
x,y = load_breast_cancer(return_X_y=True)

data = load_wine()
df = pd.DataFrame(data['data'], columns=data['feature_names'])
df['Target'] = data['target']
X = df.drop('Target', axis=1)
y = df['Target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
random_state=1376)

start_time = time.time()
svc = SVC(kernel="linear", C=0.025)
svc.n_iter_ = 100
svc.fit(X_train, y_train)
test_predictions = svc.predict(X_test)
train_predictions = svc.predict(X_train)
print("Train:")
print(classification_report(y_train, train_predictions))
print("Test:")
print(classification_report(y_test, test_predictions))
print("--- %s seconds ---" % (time.time() - start_time))

```

References

- [1] Arunachalam S and de Wolf R 2017 A Survey of Quantum Learning Theory *Preprint* arXiv:1701.06806 [quant-ph]
- [2] Cambridge Quantum Computing 2020 *Technology* URL <https://cambridgequantum.com/technology/>
- [3] Cheng C H and Wei L Y 2007 New entropy clustering analysis method based on adaptive

- learning *Proc. of the 10th Joint Conf. on Information Sciences 2007* ed P P Wang pp 1196–1202 URL https://doi.org/10.1142/9789812709677_0169
- [4] Clark J and Stepney S 2002 Quantum Software Engineering *Workshop on Grand Challenges for Computing Research* (Edinburgh: e-Science Institute) URL <http://www.ukcrc.org.uk/press/news/call/a5.cfm>
- [5] Dunjko V and Wittek P 2020 A non-review of Quantum Machine Learning: trends and explorations *Quantum Views* **4** 32 doi:10.22331/qv-2020-03-17-32
- [6] D-Wave Systems Inc 2021 *D-Wave Ocean Software Documentation* URL <https://ocean.dwavesys.com/>
- [7] Gartner 2021 Quantum Computing *Gartner Glossary* URL <https://www.gartner.com/en/information-technology/glossary/quantum-computing>
- [8] Google 2016 *Quantum Computing Playground* URL <http://www.quantumplayground.net>
- [9] Lehka L V and Shokaliuk S V 2018 Quantum programming is a promising direction of IT development *CEUR Workshop Proceedings* **2292** 76–82
- [10] Microsoft 2021 *Microsoft Quantum Documentation and Q# API Reference - Microsoft Quantum* URL <https://docs.microsoft.com/en-us/quantum/>
- [11] Microsoft Research 2016 *Language-Integrated Quantum Operations: LIQUi>* URL <https://www.microsoft.com/en-us/research/project/language-integrated-quantum-operations-liqui/>
- [12] Murali P, Linke N M, Martonosi M, Javadi-Abhari A, Nguyen N H and Alderete C H 2019 Full-Stack, Real-System Quantum Computer Studies: Architectural Comparisons and Design Insights *ISCA'19: Proc. 46th Int. Symp. on Computer Architecture* pp 527–40 URL <https://doi.org/10.1145/3307650.3322273>
- [13] Panetta K 2019 The CIO's Guide to Quantum Computing *Smarter With Gartner* URL <https://www.gartner.com/smarterwithgartner/the-cios-guide-to-quantum-computing/>
- [14] Phillipson F 2020 Quantum Machine Learning: Benefits and Practical Examples *CEUR Workshop Proceedings* **2561** 51–6
- [15] Piattini M, Peterssen G, Perez-Castillo R, Hevia J L, Serrano M A, Hernández G, de Guzmán I G R, Paradela C A, Polo M, Murina E, Jiménez L, Marqueño J C, Gallego R, Tura J, Phillipson F, Murillo J M, Niño A and Rodríguez M 2020 The Talavera Manifesto for Quantum Software Engineering and Programming *CEUR Workshop Proceedings* **2561** 1–5
- [16] Pistoia M and Gambetta J 2018 Qiskit Aqua – A Library of Quantum Algorithms and Applications *Medium* URL <https://medium.com/qiskit/qiskit-aqua-a-library-of-quantum-algorithms-and-applications-33ecf3b36008>
- [17] Qiskit 2021 *Qiskit* URL <https://qiskit.org/>
- [18] Qiskit 2021 Qiskit/openqasm: Gate and operation specification for quantum circuits *GitHub* URL <https://github.com/Qiskit/openqasm>
- [19] Q-SE2020 2020 *First International Workshop on Quantum Software Engineering (Q-SE 2020) co-located with ICSE 2020* URL <https://q-se.github.io/qse2020/>
- [20] Rahaman M and Islam M M 2015 A Review on Progress and Problems of Quantum Computing as aService (QCaas) in the Perspective of Cloud Computing *Global Journal of Computer Science and Technology: B Cloud and Distributed* **15** URL https://globaljournals.org/GJCST_Volume15/3-Cloud-Data-Storage.pdf
- [21] Rigetti Computing 2020 *Rigetti QCS* URL <https://qcs.rigetti.com/sdk-downloads>
- [22] Steiger D and Häner T 2017 *ProjectQ – Open Source Software for Quantum Computing* URL <https://projectq.ch/>
- [23] Wittek P 2016 *Quantum Machine Learning: What Quantum Computing Means to Data Mining* (San Diego: Academic Press) p 176